ELEVATE Repository Technical Run Book and Documentation

Project Overview

The ELEVATE repository, created by TeamElevate, is an online community centric platform for consumer tribes. It presents an exciting and comprehensive community app for motorcycle enthusiasts. Drawing inspiration from existing apps and community needs, here's a concept for a biker-focused app that could be developed during the hackathon

Key Features

1. **Ride Planning and Tracking**
    - GPS route planning and real-time tracking
    - Lean angle and speed monitoring
    - Offline maps for remote areas
2. **Community Engagement**
    - Social networking for bikers
    - Group ride organization and management
    - Photo and experience sharing
3. **Safety Features**
    - Crash detection and emergency alerts
    - Weather forecasts and road condition updates
    - Maintenance reminders and tracking
4. **Discovery**
    - Curated routes and points of interest
    - Local biker meetups and events
    - Motorcycle-friendly businesses
5. **MyMechanic**
    - AI chatbot for Roadside Assistance
    - It uses Retirval Augmented Generation to craft responses
    - Knowledge base consists of the User Manual, Spec sheets, FAQs etc

Technical Considerations

- Utilize open-source mapping solutions to reduce reliance on expensive APIs
- Implement offline functionality for areas with poor connectivity
- Ensure robust user privacy and data security measures
- Develop cross-platform compatibility (iOS and Android)

Potential Challenges

- Balancing feature richness with app performance and battery life
- Ensuring accurate crash detection without false positives
- Building a critical mass of users for community features to be effective

By focusing on these elements, hackathon participants can create an app that not only serves the practical needs of bikers but also fosters a strong, engaged community of motorcycle enthusiasts.

Repository Structure

The repository consists of the following key components:

- **Python Scripts**
  - `main.py`
  - `config.py`
  - `firebase_test.py`
- **Jupyter Notebook**
  - `ElevateMap.ipynb`
- **Data Files**
  - `matches_user3.json`
  - `matches_user5.json`
- **Directories**
  - `__pycache__`
  - `data`
  - `modelWrappers`
  - `rag_docs`
  - `utils`

Detailed File Contents

Python Scripts

1. `main.py`
   - **Purpose**: This is the entry point of the application. It orchestrates the execution of the main functionalities.
   - **Key Features**:
     - Imports necessary modules and packages.
     - Initializes configurations from `config.py`.
     - Handles user input or command-line arguments.
     - Calls functions from other modules to execute core features.
2. `config.py`
   - **Purpose**: Contains configuration settings for the application.
   - **Key Features**:
     - Defines constants such as API keys, file paths, and other environment-specific settings.
     - Facilitates easy adjustments to configurations without modifying core logic.
3. `firebase_test.py`
   - **Purpose**: Tests Firebase integration.
   - **Key Features**:
     - Connects to a Firebase project using credentials defined in the environment.
     - Contains functions to read/write data to Firebase, useful for backend operations.

Jupyter Notebook

- `ElevateMap.ipynb`
  - **Purpose**: Provides an interactive environment for data visualization and analysis.
  - **Key Features**:

    - Visualizes data from JSON files using libraries like Matplotlib or Seaborn.
    - Allows for exploratory data analysis (EDA) through code cells that can be executed independently.

Data Files

- `matches_user3.json` & `matches_user5.json`
    - **Purpose**: Sample data files used within the application.
    - **Key Features**:
        - Contain structured data that can be loaded into the application for processing or analysis.
        - Typically represent user match data, which could be used for machine learning or analytics.

Directories

1. `__pycache__`
    - Contains compiled Python files (.pyc) for performance optimization. This directory is automatically generated by Python and does not require manual intervention.
2. `data`
    - Intended to store additional datasets or resources required by the application.
3. `modelWrappers`
    - Contains wrappers for machine learning models or APIs.
    - Each file in this directory likely encapsulates functionality related to specific models, making it easier to integrate them into the main application.
4. `rag_docs`
    - May contain documents related to research, architecture, or guidelines pertinent to the project.
5. `utils`
    - A collection of utility functions that support various tasks within the application (e.g., data preprocessing, logging).

Requirements

To run the ELEVATE project successfully, ensure you have the following:

- Python version: Check compatibility (likely Python 3.x).
- Required libraries: Install dependencies listed in a `requirements.txt` file (if available) using:

```
pip install -r requirements.txt
```

Common libraries may include:

- pandas
- numpy
- matplotlib
- seaborn
- firebase-admin

Features and Significance

- **Modular Design**: The separation of concerns through different files and directories allows for easier maintenance and scalability of the codebase.
- **Interactive Analysis**: The use of Jupyter Notebooks enables users to visualize and analyze data interactively, which is crucial for exploratory data analysis.
- **Firebase Integration**: The ability to connect with Firebase allows for real-time database capabilities, enhancing the application's interactivity and responsiveness.

Running the Application

1. Clone the repository:

```
git clone https://github.com/KumarShivam1908/ELEVATE.git
cd ELEVATE
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Run the main application:

```
python main.py
```

4. For Jupyter Notebook usage:

```
jupyter notebook ElevateMap.ipynb
```

Configuration Settings

Modify `config.py` to customize settings such as:

- Firebase credentials
- File paths for data input/output
- API keys if applicable

Contributing Guidelines

To contribute to this repository:

1. Fork the repository on GitHub.
2. Create a new branch for your feature or bug fix.
3. Make your changes and commit them with clear messages.
4. Push your branch and create a pull request detailing your changes.

Conclusion

This technical run book provides an in-depth overview of the ELEVATE repository's structure, contents, requirements, features, and significance. For further exploration, it is recommended to delve into individual files and their respective documentation within the codebase.